UNITED STATES PATENT AND TRADEMARK OFFICE

[illegible]

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/827,739	04/06/2001	John H. Sandham	1801270.00121US1	6702

23483 7590 09/18/2006

WILMER CUTLER PICKERING HALE AND DORR LLP
60 STATE STREET
BOSTON, MA 02109

EXAMINER	
PROCTOR, JASON SCOTT	
ART UNIT	**PAPER NUMBER**
2123	

DATE MAILED: 09/18/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/827,739	SANDHAM, JOHN H.
	Examiner	Art Unit
	Jason Proctor	2123

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 June 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4-10,12,14 and 15* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4-10,12,14 and 15* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

Claims 1-12 were rejected in the Office Action of 21 December 2005. Applicants' response dated 19 June 2006 has amended claims 1, 2, and 4; cancelled claims 3 and 11; and presented new claims 14 and 15.

Claims 1-2, 4-10, 12, and 14-15 are pending in this application. Claims 1-2, 4-10, 12, and 14-15 are rejected.

Claim Rejections – 35 USC § 101

1. The previous rejections of claims 1-4 under 35 U.S.C. § 101 have been withdrawn in response to the amendments to those claims.

Claim Rejections – 35 USC § 112

2. Claims 2 and 10 are rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. Claim 2 recites steps that are not operative for the stated purpose of "use in emulating a processor". Claim 10 recites a system that employs the method of claim 2 and fails to provide an "endian transformation system" as would be recognized by a person of ordinary skill in the art.

Making and/or using the invention of claims 2 and 10, which substantially recite creating an inverted memory image, where "the offset between addresses of any two bytes stored in memory is unaltered by the transformation, wherein said any two bytes are spaced apart in memory by a plurality of words, and the relative order of the addresses of said any two bytes stored in the memory is reversed by the transformation" would require numerous inventions and modifications to the "second type of processor," such as but not limited to a decrementing program counter, a novel implementation of numerical data storage in memory, and a novel implementation of pointer arithmetic.

In every computer processor known to the Examiner, a program counter is automatically incremented to the next higher instruction address upon completion of an instruction. If the program instructions were reversed in memory, according to the method of claim 2, the computer processor would read the program instructions in the opposite order that they were intended. The automatic increment of a program counter is implemented in the hardware of a computer processor and not readily modified or adapted.

Similarly, both a big endian and little endian computer system share numerous properties related to memory addressing. It is only when storing data *within words* that they differ. Reversing the contents of an entire memory space, that is, *reversing words and their contents* would interfere with normal execution of both types of processors, especially regarding numerical data storage in memory and pointer arithmetic. The same arguments apply to the system of claim 10.

In response, Applicants' argue primarily that:

> However, the method of claim 2 is for use in emulating a first processor using a second processor. Such emulation involves the transformation of program code from the first processor (subject system) to run on the second processor (target system). In generating the target instructions, the transformation process has

> complete control over the manner in which those instructions will be executed. The Examiner correctly notes that the hardware of the target system will automatically increment the target program counter. The claimed method concerns the effect of executing those target instructions – namely that the memory access addresses provided in the subject instructions are transformed as claimed. As the application makes clear, such transformed memory access addresses apply to a plurality of bytes within memory as appropriate to the subject program. Hence, the target program counter can indeed automatically increment as is usual for the target hardware, but still the memory addresses affected by executing those target instructions address the reverse relative order of words as claimed. The "inventions and modifications" to the second processor suggested by the Examiner are not necessary. (emphasis in original)

The Examiner respectfully traverses this argument as follows.

Claim 2 does not appear to recite "a transformation process [that] has complete control over the manner in which instructions will be executed." The Examiner is unaware of support under 35 U.S.C. § 112, first paragraph, in the specification for this argument. It appears that Applicants' argument may rely upon the fact that the claim fails to define any steps of transformation, but instead merely describes the result of the transformation ("transforming... such that..."). However, while this claim language is extremely broad, there is no indication in the claim language itself that the transformation process "has complete control over the manner in which instructions will be executed." In contrast to the transformation process having complete control, a reasonable person of skill in the art would recognize that the transformed instructions would be executed according to the constraints imposed by the processor of the second type.

Secondly, Applicants' argument that "transformed memory access addresses apply to a plurality of bytes within memory as appropriate to the target hardware" is not understood. It is unclear if this is a suggestion that the claim should be interpreted as broadly as the enablement supports. This interpretation would be improper because the claims must be given a broad and reasonable interpretation during examination as per MPEP 2111.

Regardless, the claim calls for transforming memory access addresses and makes no apparent distinction between memory addresses storing data and memory addresses storing the instructions to be executed. Instructions to be executed are stored in sequential ascending order in every processor known to the Examiner. The program counter increments to the next memory address after executing an instruction to point to the next sequential instruction. The claimed invention disrupts this arrangement by producing program instructions in *descending* sequential order and provides no alternative teachings that enable implementation of the method.

Inasmuch as the sequential access of instructions is somehow excluded by the phrase "memory access addresses," the same cannot be said for jump and branch instructions that explicitly specify a destination "memory access address".

Applicants' arguments have been fully considered but have been found unpersuasive.

3. Claims 1-2 and 4 are rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. The specification as filed does not describe "issuing instructions for execution by said processor of the second type including said transformed memory access addresses."

4. Claim 15 is rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that

the inventor(s), at the time the application was filed, had possession of the claimed invention. The specification as filed does not describe "calculating variable terms of the expression A-B-L+S to provide fully resolved transformed memory access addresses."

The following is a quotation of the second paragraph of 35 U.S.C. § 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claim 14 is rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 14 recites "transforming said memory access addresses into a respective transformed memory access address A-B-L+S, such that the relative order of bytes within each word is reversed into the second endian format and the plurality of words are addressed in a second relative order with respect to the given starting address which is a mirror image reverse of the first relative order" which renders the claim vague and indefinite.

"Transforming said memory access addresses" alone <u>cannot</u> "[reverse] the relative order of bytes within each word ... into the second endian format." Memory access addresses are references or labels for physical memory locations. Transforming those references does not alter the contents of the physical memory locations. By way of analogy, the claim recites changing a person's phone number in a telephone book (transforming a reference) and thereby altering the real-life phone number. Therefore the claim omits a necessary step of actually manipulating the memory contents to achieve the claimed result.

The claim language "transforming said memory access addresses ... such that ... the plurality of words are addressed in a second relative order with respect to the given starting address which is a mirror image reverse of the first relative order" apparently describes the intended use for the plurality of words in random access memory. In the context of the claimed invention, this language appears to seek patent protection for the particular computer programs wherein "the plurality of words are addressed" in a particular order. The scope of invention described by this language is unclear.

The claim language "to enable code instructions adapted for execution on a first type of processor which observes a first endian format for ordering the significance of bytes within words to be executed by a second type of processor which observes a second endian format for ordering the significance of bytes within words" and similarly "translating the code instructions into output code instructions executable by the second type of processor, where said output code instructions include said transformed memory access addresses"

6. Claim 15 is rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 15 does not appear to further limit the method steps defined by its parent claim 14. Claim 14 defines, *inter alia*, a translation phase that produces "output code instructions [which] include said transformed memory access addresses" and an execution phase that executes those instructions using the transformed memory access addresses.

In contrast, claim 15 apparently broadens the translation phase by reciting that the translation phase produces “partially resolved transformed memory access addresses in the translated output code instructions” (rather than fully resolved memory access addresses).

Further, the execution phase of claim 14 recites “executing said output code instructions … using the transformed memory access addresses,” whereas claim 15 recites a step producing “fully resolved transformed memory access addresses.” Presuming that “transformed memory access addresses” are equivalent to “fully resolved transformed memory access addresses,” claim 15 attempts to rearrange the limitations of claim 14 by producing the “transformed memory access addresses” in the execution phase rather than the translation phase. Claim 15 defines a different invention, not a further limitation of the invention in claim 14.

The claim language “in the translation phase, folding constant terms in the expression A-B-L+S to provide partially resolved transformed memory access addresses in the translated output code instructions” is vague and indefinite for several reasons. The meaning of the term “folding” is unclear. The phrase “constant terms” lacks antecedent basis. The parent claim recites “A-B-L+S” as a description of the transformed address, **not** as a process of arithmetic operation. Therefore the meaning of “partially resolved transformed memory access addresses” in the context of the claims is unknown.

The claim language “in the execution phase, calculating variable terms of the expression A-B-L+S to provide fully resolved transformed memory access addresses” is vague and indefinite for several reasons. The phrase “variable terms” lacks antecedent basis. It is unclear how “calculating variable terms” can be interpreted as providing “fully resolved transformed memory access addresses." This language does not appear to account for the "folded constant terms" recited previously in the claim.

In summary, claim 15 is so indefinite that no prior art examination is feasible. Specifically, the Examiner should not rely "on what at best are speculative assumptions as to the meaning of the claims", and should not base "a rejection under 35 U.S.C. 103 thereon...[when] the claims do not particularly point out and distinctly claim the invention as required by 35 U.S.C. 112." *In re Steele*, 305 F.2d 859, 134 USPQ 292, 295 (CCPA 1962). Also see *In re Citron*, 45 CCPA 773, 251 F.2d 619, 116 USPQ 409.

Claim Rejections - 35 USC § 102

The previous rejections of claims 2 and 10 under 35 U.S.C. § 102 have been withdrawn.

The following is a quotation of the appropriate paragraphs of 35 U.S.C. § 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

7. Claims 1, 4-9, and 12 are rejected under 35 U.S.C. § 102(e) as being anticipated by US Patent No. 5,968,164 to Loen et al. (Loen).

Regarding claim 1, Loen discloses receiving instructions for execution by said processor of a first type, wherein the instructions comprise memory access addresses in which a byte order in accordance with the first convention is observed [*ex.* FIG. 1, 155, "Big Endian Programs"; 177, "Big Endian Data"];

Transforming the memory access addresses for an entire address space over a plurality of words such that bytes stored in a memory addressed by the processor of the second type as a result of the instructions are distributed in a pattern which is a mirror image of the distribution pattern of the bytes which would result if the memory was addressed by the processor of the first type in response to the said instructions [*"When the circuitry is initialized to operated in the alternate endian mode, reflection of the data takes place... The second step, called address modification, converts the addresses used to reference the data from big endian addresses to little endian addresses..."* (column 6, line 20 – column 7, line 29); FIGS. 3A-3B]; and

Issuing instructions for execution by said processor of the second type including said transformed memory access addresses [*"...the present invention is an enhanced computer system that allows tasks, having different endian expectations (i.e., either big or little) to not only co-exist on a single computer system, but to execute, task for task, on a single computer system as well."* (column 4, lines 49-55)].

In response, Applicants argue primarily that:

Loen provides a specially adapted form of hardware which includes "reflection circuitry" 121 of Figure 1, which reverses the order of bytes within a word during a fetch from memory. Loen performs a modification of the address offset within a word using the hardware offset modification circuitry 119 of Figure 1, so that the address offset still points to the correct byte within the word, even though the order of bytes within the word has been reflected. That is, the hardware of Loen transforms byte order within a data word, and adjusts a memory address offset on a word by word basis as part of the fetch and store pathway. Loen does not receive instructions, transform memory access addresses with reference to those instructions, and then issue instructions for execution by the second processor, where those issued instructions include the transformed memory access addresses, as claimed.

In summary, Loen does not transform memory access addresses with reference to instructions, but instead applies a hardware transformation to data words within the fetch and store pathway. Further, Loen

> only manipulates the bytes within a data double word or part thereof. Loen does not rearrange the words, but the transformation of the claimed invention does. (emphasis in original)

The Examiner respectfully traverses this argument as follows.

Loen discloses (column 10, lines 24-28):

> Those skilled in the art will recognize that it is possible to implement reflection circuitry 121 in software running on the cache memory controller associated with cache memory 103 (not shown) without loss of generality.

Additionally, the claim does not clearly specify that a hardware transformation would be excluded from the claim scope.

Further, the claimed invention <u>does not rearrange the words</u>. The claim recites "transforming the memory access addresses for an entire address space." A reasonable person of ordinary skill in the art would recognize that *transforming the addresses*, i.e. transforming the labels referring to memory locations, is distinctly different than *transforming the words*, i.e. manipulating the contents of the memory locations. Claim 1 does not recite "rearranging the words" or any reasonable equivalent.

The claim recites that "bytes stored in a memory addressed by the processor of the second type as a result of the instructions are distributed in a pattern which is a mirror image of the distribution pattern of the bytes **<u>which would result if</u>** the memory was addressed by the processor of the first type in response to the said instructions". This language does not clearly specify the step of "transforming". This language is at least open to the interpretation that none of the instructions store bytes. The meaning of the term "mirror image" is also open to interpretation.

In summary, claim 1 defines the process of "transforming" merely by describing the **results** of the process. Claim 1 does not positively recite any identifiable steps of "transforming". Therefore, attempts to distinguish the claimed invention on the basis of the process of transforming are ineffective. The prior art of record discloses a process of "transforming" that meets the claim language.

Applicants' arguments have been fully considered but have been found unpersuasive.

Regarding claim 4, Loen discloses receiving instructions for execution by said processor of a first type, wherein the instructions comprise memory access addresses in which a byte order in accordance with the first convention is observed [*ex.* FIG. 1, 155, "Big Endian Programs"; 177, "Big Endian Data"];

The memory address B of word length L is transformed to the address A-B-L+S where A is the total number of bytes allocated to the program and S is the start address of the program (Figures 4A-4D). Figures 4A-4D of Loen et al. discloses an example where A=8, S=0, and L is 8, 16, 32, or 64 bit, however also teaches other values for L (column 7, lines 62 – column 8, line 46); and

Issuing instructions for execution by said processor of the second type including said transformed memory access addresses [*"...the present invention is an enhanced computer system that allows tasks, having different endian expectations (i.e., either big or little) to not only co-exist on a single computer system, but to execute, task for task, on a single computer system as well."* (column 4, lines 49-55)].

For example, FIG. 4B shows the recalculated offset to access a halfword (2 bytes) after reflecting the memory. In this case, the address being accessed is B=4. The length of the data being accessed is L=1. The new offset as shown in FIG. 4B is A-B-L+S = 8-4-2+0 = 2.

In response, Applicants argue primarily that:

> [In Loen, T]he byte reflection and address offset modification are applied to each word individually in turn. Loen does not provide a transformation that applies to a plurality of words. We submit that the language "relating to a plurality of words is sufficient to bring out this distinction between performing a task individually but repeated a number of times, compared with performing a task in relation to a whole group.

The Examiner respectfully traverses this rejection as follows.

The Examiner acknowledges Applicants' position but maintains that the explicit claim language "transforming a plurality of the memory access addresses relating to a plurality of words" does not exclude transforming a series [a plurality] of individual memory access addresses.

Further, the Examiner does not understand the application to disclose an atomic computer instruction that is capable of transforming multiple addresses in parallel or simultaneously. That is, the Examiner understands that the invention, at certain level of scrutiny, transforms memory access addresses as a series of individual transformations. The Examiner is skeptical that the alternative, a general computer operation that transforms a plurality of addresses in a single step or operation, is supported under 35 U.S.C. § 112, first paragraph. The claimed invention is not limited to, for example, a parallel processing system or a systolic array processor, either of which may be ordinarily capable of performing an atomic operation on a plurality of data items.

Lastly, the Examiner submits that the claim describes the process of "transforming" merely by describing the **results** of the process. Therefore, attempts to distinguish the claimed invention on the basis of the process of transforming are ineffective. The prior art of record discloses a process of "transforming" that meets the claim language.

Applicants' arguments have been fully considered but have been found unpersuasive.

Regarding claim 5, Loen et al. discloses a process for translating a program code instruction for execution on a programmable machine using a corresponding predetermined convention of ordering the significance of bytes within words of the address space (column 6, line 20 – column 7, line 29; Figures 3A-3B) comprising:

transforming the referenced memory address with respect to a fixed block size of memory in the programmable machine so as to change the referenced address value by an amount that is fixed for a given number of bytes being accessed in each word (Figures 4A-4D; column 7, line 62 – column 8, line 46);

including the changed address reference in the output instruction so that there is no extra operation required during execution of the instruction to accommodate the convention for ordering bytes within words used by said predetermined programmable machine (column 7, line 62 – column 8, line 46).

In response, Applicants argue primarily that:

> … Loen does not disclose "including the thus changed address reference in a compiled or translated output instruction" as claimed… Loen reflects the data within a particular word, when that word is fetched into the processor. Loen does not include changed address references in a compiled or translated output instruction as in claim 5. Further, claim 5 recites that the changed address reference is included "such that there is no extra operation required during execution of the output instruction to accommodate the [alternate endian] convention". In claim 5, this feature provides a clear distinction over Loen where the reflection circuitry 121 and the address offset modification circuitry 119 are clearly required during execution of the instruction.

The Examiner respectfully traverses this argument as follows.

The Examiner first observes that the claim refers in several instances to "during compilation or translation of a code". The prior art may show either compilation or translation to anticipate the claim.

To address Applicants' first point, Loen includes a changed address reference in a translated output instruction as in claim 5. See Loen, column 7, line 62 – column 8, line 46, in particular, *"To complete the processor fetch from memory, an address modification is performed on the address as originally presented by the software"* (column 7, lines 65-67).

To address Applicants' second point, the claim language "such that there is no extra operation required during execution of the output instruction to accommodate the [alternate endian] convention" merely describes the result of "including" rather than positively identifying the method steps that could help to distinguish the invention over the prior art. Further, the language "no extra operation required" is clearly open to several interpretations. Regardless, the hardware circuitry of Loen enables the invention to operate with no extra computer code operation to accommodate the alternate endian convention, because the accommodation is made in the hardware circuitry. Loen may anticipate several other interpretations of the claim language.

Lastly, the Examiner submits that the claim describes the process of "transforming" merely by describing the results of the process. Therefore, attempts to distinguish the claimed invention on the basis of the process of transforming are ineffective. The prior art of record discloses a process of "transforming" that meets the claim language.

Applicants' arguments have been fully considered but have been found unpersuasive.

Regarding claim 6, Loen et al. discloses that the system is applied to program source code (column 7, lines 62 – column 8, line 46).

Regarding claim 7, Loen et al. discloses that a fixed block of memory is either big endian or little endian and therefore addressed from a predetermined one of its two ends depending upon the convention utilized for ordering the significance of bytes within the words (column 6, line 20 – column 7, line 29; Figures 3A-3B).

In response, Applicants argue primarily that:

Loen does not disclose this step of determining which of the available two ends should be utilized.

The Examiner respectfully traverses this argument as follows.

In Loen, the memory is addressed in a conventional method. That is, it is determined to address the memory from the "low" end starting with address 0. The claim language requires nothing further.

Applicants' arguments have been fully considered but have been found unpersuasive.

Regarding claim 8, Loen et al. discloses that the translation causes a fixed block of memory contents for a big-endian machine to be inverted to the mirror image of that for a little-endian machine (column 6, line 20 – column 7, line 29; Figures 3A-3B).

Claims 9 and 12 recite "an endian transformation system" which performs the methods of claims 1 and 4, respectively. As the invention disclosed by Loen et al. is indeed a system that performs a method used to reject claims 1 and 4 (column 4, lines 49-55), claims 9 and 12 are rejected for the same reasons used to reject claims 1 and 4 above.

The following is a quotation of the appropriate paragraphs of 35 U.S.C. § 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claim 14 is rejected under 35 U.S.C. 102(b) as being anticipated by "The PowerPC User Instruction Set Architecture" by Keith Diefendorff (hereafter referred to as Diefendorff).

Regarding claim 14, Diefendorff discloses:

Allocating a memory address range of length A bytes comprising a plurality of words arranged in a first relative order with respect to a starting address S [FIG. 7(b) page 35, memory address 14 through 20, storing character array *"char d[7]"* from page 34, right column];

Receiving code instructions having memory access addresses which address the memory address range according to the first endian format for ordering the significance of bytes within words, where each access address B is of string length L [*"Load and store instructions access a memory-quantum size equal to the size of their data type. For example, a load-byte instruction reads one byte from memory; a store byte stores one byte in memory."* (page 38, left column); here "Load byte" provides an address B of string length L=1];

Transforming said memory access addresses into a respective transformed memory access address A-B-L+S [*"One simple way for processors to implement little-endian capability* *is to exclusive-OR (XOR) a few low-order bits of the physical memory address, with values that depend upon the size of the data type being accessed. ... we XOR the bottom three bits of the address with 7 for byte accesses...* " (page 34, right column)];

Here Diefendorff discloses a little-endian storage of " GFEDCBA" from left to right in memory addresses 21-14 (FIG 7(a)). This discloses to a person of ordinary skill in the art that a struct { char d[7]} will be stored in a little-endian convention as " GFEDCBA" from left to right in memory addresses 8-0. This also discloses to a person of ordinary skill in the art that storing a longer array of bytes will occupy a corresponding larger number of memory locations according to the depicted convention.

Diefendorff discloses the big-endian storage of the same data, "ABCDEFG " from left to right in memory addresses 10-21 (FIG 7(b)). This discloses to a person of ordinary skill in the art that a struct { char d[7]} will be stored in big-endian convention as "ABCDEFG " from left to right in memory addresses 0-8. This also discloses to a person of ordinary skill in the art that storing a longer array of bytes will occupy a corresponding larger number of memory locations according to the depicted convention.

Diefendorff depicts the storage of the byte array in combination with other data items which are not required by the claim nor would a person of ordinary skill in the art believe they are necessary in practice. Diefendorff depicts the memory address numbering scheme native to each convention which is not a component of the claimed invention.

Thus Diefendorff discloses to a person of ordinary skill in the art the storage of the string " GFEDCBA" in (renumbered) memory locations 0-8 in a little-endian convention and, equivalently, "ABCDEFG " in memory locations 0-8 in a big-endian convention.

Here the address space A=8, the start address S=0, and loading a byte has a length L=1. By way of example, memory access address B = 5 (character "C") in little-endian storage is transformed to A – B – L + S = 8 – 5 – 1 + 0 = 2 in big-endian storage. See illustration below.

```
0 1 2 3 4 5 6 7

_ G F E D C B A

A B C D E F G _
```

Further, Diefendorff discloses performing subtraction to transform the address [*"We could replace the XOR with a subtraction operation to support misaligned references within doubleword boundaries. Subtraction is slower, however, and the computation required across the doubleword boundary is even more complex. But the architecture does allow this implementation if necessary."* (page 35, left column)];

Translating the code instructions into output code instructions executable by the second type of processor, where said output code instructions include said transformed memory access addresses [*"With this, we can simply recompile (as opposed to port) programs created for little-endian machines and run them on PowerPC processors."* (page 35, left column)]; and

Executing said output code instructions on said second type of processor to fetch and store data in the plurality of words in the memory address range using the transformed memory access addresses [Table 3, page 38, "Load and store instructions"; *"The operation of most of the instructions in Table 3 is straightforward and self explanatory. The load and stores with byte-*

reverse swap individual bytes end-for-end within their indicated data type on the way to or from memory. This feature provides support of mixed-endian systems." (page 38, right column)].

The Examiner submits that the claim does not recite *performing* the operation "A-B-L+S", but rather merely uses this notation to describe what the resulting transformed address will be. The claim language "such that... first relative order" describes the step of "transforming" but presents no clearly identifiable method steps. Therefore, where the prior art produces an equivalent transformed address, that prior art is interpreted as anticipating the claim language "such that the relative order of bytes within each word is reversed into the second endian format and the plurality of words are addressed in a second relative order with respect to the given starting address which is a mirror image reverse of the first relative order."

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. § 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* v. *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. § 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.

2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

9. Claims 2 and 10 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "Data Structures & Other Objects Using C++" by Michael Main and Walter Savitch (Savitch) in view of US Patent No. 5,968,164 to Loen et al. (Loen).

Regarding claim 2, Savitch discloses a method comprising transforming memory access addresses such that (a) the offset between addresses of any two bytes stored in memory is unaltered by the transformation, wherein said any two bytes are spaced apart in memory by a plurality of words, and (b) the relative order of the addresses of said any two bytes stored in the memory is reversed by the transformation (Chapter 7 Stacks; particularly "Programming Example: Reversing a Word", pages 310-311).

Savitch describes a method where a word (here referring to an English language word, using "NAT" as exemplary input) is manipulated to produce "TAN". As would be well known to a person of ordinary skill in the art, character data is typically stored as one character corresponding to one byte. Thus Savitch discloses a sequence of three bytes storing N, A, and T, respectively, being reversed such that they store T, A, and N. This transformation clearly meets requirements (a) and (b) as set forth in the claim. Savitch provides, in lesser detail, a longer example of the input data "ESIOTROT" being transformed to "TORTOISE" (page 310).

Savitch does not disclose receiving instructions for execution by a processor of a first type and issuing instructions for execution by a processor of a second type.

Loen discloses receiving instructions for execution by said processor of a first type, wherein the instructions comprise memory access addresses in which a byte order in accordance with the first convention is observed [*ex.* FIG. 1, 155, "Big Endian Programs"; 177, "Big Endian Data"]; and

Issuing instructions for execution by said processor of the second type including said transformed memory access addresses [*"... the present invention is an enhanced computer system that allows tasks, having different endian expectations (i.e., either big or little) to not only co-exist on a single computer system, but to execute, task for task, on a single computer system as well."* (column 4, lines 49-55)].

Loen teaches that the reflection circuitry could be implemented in software [*"Those skilled in the art will recognize that it is possible to implement reflection circuitry 121 in software running on the cache memory controller associated with cache memory 103 (not shown) without loss of generality."* (column 10, lines 20-36)].

Savitch and Loen are analogous art because both are drawn to computer software.

It would have been obvious to a person of ordinary skill in the art at the time of Applicants' invention to use a data structure taught by Savitch to implement the "reflection circuitry" as suggested by Loen. The example shown in Savitch performs this function.

Motivation to do so is expressly taught in Loen, such as a computer system that can execute tasks having different endian expectations (Loen, column 4, lines 49-55).

Therefore it would have been obvious to a person of ordinary skill in the art to combine the teachings of Savitch and Loen in order to implement the system taught by Loen, as specified in claim 2.

In response, Applicants argue primarily that:

First, taking the Examiner's analogy that one letter equals one byte, the stack of Savitch only reverses the relative order of bytes (letters) within a word. Savitch does not disclose reversing the order of words.

The Examiner respectfully traverses this argument as follows.

Where Applicants' argument may refer to the grammatical definition of *word*, Savitch most certainly teaches reversing the order of words. A person of ordinary skill in computer programming would immediately recognize that the data structure and algorithm taught by Savitch would transform the phrase "two words" into "sdrow owt", which clearly reverses the order of the words.

Where Applicants' argument may refer to the computer technology definition of *word*, which is usually defined as two or four bytes on most computer systems, Savitch most certainly teaches reversing the order of words. In Savitch's explicit example, the input "TORTOISE" comprises 4 2-byte words. Specifically, these are "TO" "RT" "OI" and "SE". Savitch expressly discloses that the transformed output is "ESIOTROT" wherein the 4 2-byte words are "ES" "IO" "TR" and "OT". Savitch has not only reversed the bytes within words ("TO" becomes "OT") but has also reversed the order of the words (the first word becomes the last word).

Applicants further argue that:

Secondly, there is no motivation to combine the teachings of Savitch with the teaching of Loen, since combing the stack of Savitch with the hardware of Loen is illogical. Savitch concerns "data structures and other objects using C++" as a high level programming language. Claims 2 and 10 concern transforming memory access addresses within instructions executable by a processor – i.e., assembly code or byte code level instructions. There is a world of difference between the scope and content of Savitch and the environment relating to claims 2 and 10. Even if the skilled person were able to make the huge leap from C++ programming to memory address transformation for executable instructions, Savitch still teaches no more than using a stack to reverse the relative order of bytes (letters) within a particular word, whereas claims 2 and 10 require "where said any two bytes are spaced apart in memory by a plurality of words".

The Examiner respectfully traverses this argument as follows.

Applicants' alleged "huge leap" between C++ and executable instructions is known in the art as "compiling code."

Microsoft Computer Dictionary, Fifth Edition, defines:

> **compile** *vb*. To translate all the source code of a program from a high-level language into object code prior to execution of the program. Object code is executable machine code or a variation of machine code. More generally, *compiling* is sometimes used to describe translating any high-level symbolic description into a lower-level symbolic or machine-readable format. A program that performs this task is known as a *compiler*. *See also* compiler (definition 2), compile time, high-level language, machine code, source code. *Compare* interpret.

Applicants' subsequent argument that Savitch fails to teach reversing bytes in a plurality of words appears to confuse the grammatical definition and the computer technology definition of the term *word*. This argument has been addressed above.

Applicants' arguments have been fully considered but have been found unpersuasive.

Regarding claim 10, which recites a system for performing the method of claim 2, Savitch discloses concepts of computer programming (pages 306-307) and therefore implicitly discloses that a conventional computer system would perform the method. Claim 10 is therefore rejected for the same rationale given above regarding claim 2.

Conclusion

10. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Proctor whose telephone number is (571) 272-3713. The examiner can normally be reached on 8:30 am-4:30 pm M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Paul Rodriguez can be reached at (571) 272-3753. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Jason Proctor
Examiner
Art Unit 2123

jsp

[illegible]
PAUL RODRIGUEZ
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

9/14/06